Components

Improtant!!

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML. Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.

A React component is a small, reusable code, which is responsible for one part of the application UI. A React application is an aggregation of components. React can help us to build reusable components. The following diagram shows different components. All the components have different border colors. In React we assemble different components together to create an application. We use JavaScript functions or classes to make components. If we use a function, the component will be a functional component, but if we use a class, the component will be a class-based component.

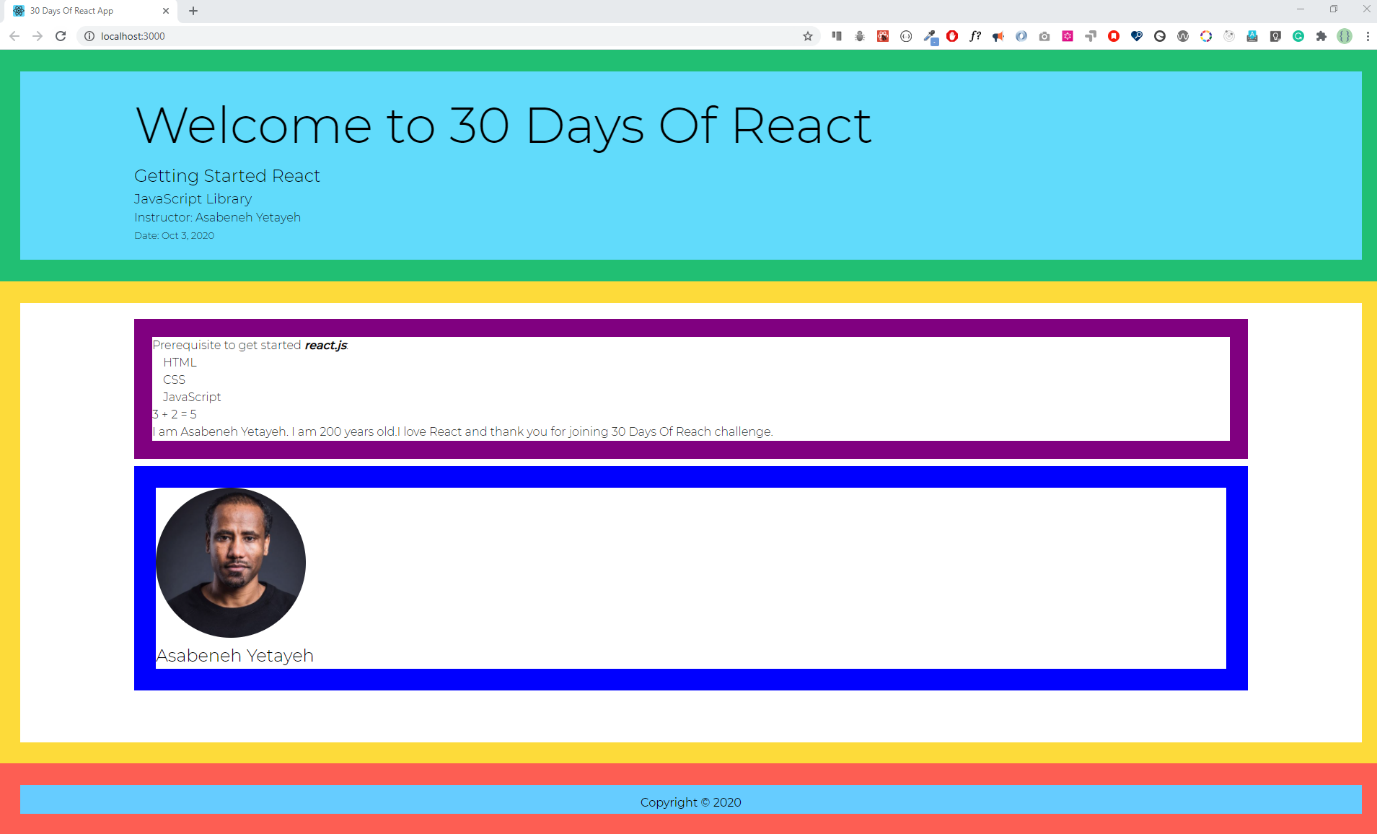
**Components can be:**

* Functional Component / Presentational Component / Stateless Component / Dumb Component
* Class Component / Container Component/ Statefull Component / Smart Component

The classification of components above does not work for the latest version of React, but it is good to know the former definition and how the previous versions work.

So, let us change all the JSX to components. Components in React are JavaScript functions or classes, that return a JSX. Component name must start with an uppercase, and if the name is two words, it should be CamelCase - a camel with two humps.

the header, main and footer are **components**. Inside the main there is also a user card component and a text section component. All the different colors represent different components. How many colors do you see? Each color represent a single component. We have five components in this diagram.



**Examples for js function and class**

JavaScript function

A JavaScript function could be either a regular function or an arrow function. These functions are not exactly the same there is a slight difference between them.

const getUserInfo = (firstName, lastName, country, title, skills) => {

return `${firstName} ${lastName}, a ${title} developer based in ${country}. He knows ${skills.join(

' '

)} `

}

// When we call this function we need parameters

const skills = ['HTML', 'CSS', 'JS', 'React']

console.log(

getUserInfo('Asabeneh', 'Yetayeh', 'Finland', 'FullStack Developer', skills)

)

JavaScript Class

A class is a blueprint of an object. We instantiate a class to create different objects. In addition, we can create children, by inheriting all the methods and properties of the parent.

class Parent {

constructor(firstName, lastName, country, title) {

// we bind the params with this class object using this keyword

this.firstName = firstName

this.lastName = lastName

this.country = country

this.title = title

}

getPersonInfo() {

return `${this.firstName} ${this.lastName}, a ${this.title} developer base in ${this.country} `

}

parentMethod() {

// code goes here

}

}

const p1 = new Parent('Asabeneh', 'Yetayeh', 'Finland', 'FullStack Developer')

class Child extends Parent {

constructor(firstName, lastName, country, title, skills) {

super(firstName, lastName, country, title)

this.skills = skills

// we bind the child params with the this keyword to this child object

}

getSkills() {

let len = this.skills.length

return len > 0 ? this.skills.join(' ') : 'No skills found'

}

childMethod() {

// code goes here

}

}

const skills = ['HTML', 'CSS', 'JS', 'React']

const child = new Child(

'Asabeneh',

'Yetayeh',

'Finland',

'FullStack Developer',

skills

)